Lalli *et al. BMC Bioinformatics* (2025) 26:248
https://doi.org/10.1186/s12859-025-06244-8

BMC Bioinformatics

SOFTWARE Open Access

JINet: easy and secure private data analysis for everyone

Giada Lalli[1*], James Collier[2*], Yves Moreau[3] and Daniele Raimondi[3,4]

*Correspondence:
Giada Lalli
giadalalli@gmail.com
James Collier
james.collier@vib.be
[1]BIO3 – Systems Medicine Lab, KU Leuven, Leuven, Belgium
[2]VIB, Ghent, Belgium
[3]ESAT-STADIUS, KU Leuven, Leuven, Belgium
[4]Institut de Génétique Moléculaire de Montpellier, Université de Montpellier, Montpellier, France

Abstract

Background The barriers to effective data analysis are sometimes insurmountable. Concerns ranging from privacy, security, and complexity can prevent researchers from using existing data analysis tools.

Results JINet is a web browser-based platform intended to democratise access to advanced clinical and genomic data analysis software. It hosts numerous data analysis applications that are run in the safety of each User's web browser, without the data ever leaving their machine.

Conclusions JINet promotes collaboration, standardisation and reproducibility by sharing scripts rather than data and creating a self-sustaining community around it in which Users and data analysis tools Developers interact thanks to JINet's interoperability primitives.

Keywords Privacy, Data-analysis, Ease-of-use, WebAssembly, Interoperability

Background

Advancing healthcare research, fostering innovation, and enhancing reproducibility and transparency in science requires comprehensive access to datasets and effective analysis tools, as well as data privacy and security measures. In healthcare, these elements are crucial for uncovering disease mechanisms, developing personalised treatments, and promoting collaborative problem-solving.

Currently, significant barriers hinder these goals. Limited access to datasets and inadequate analysis tools impede meaningful insights. Concerns over data privacy and security discourage collaboration and hamper data sharing. Additionally, interoperability challenges and disparate data formats complicate seamless integration and comprehensive analysis. Technical barriers and an exclusive research environment further restrict accessibility, especially for non-experts.

Various solutions have been proposed to overcome these challenges. Centralised repositories, workflows and cloud-based platforms, such as blockchain-based solutions [1, 2], Galaxy [3], Apache Airflow [4], Nextflow [5], and Huggingface [6, 7], offer avenues for collaboratively storing, processing, and sharing large datasets, typically involving commitments such as funding, time, and trust. However, these solutions

have notable limitations that impact their effectiveness and widespread adoption. For instance, Apache Airflow relies heavily on third-party web services, which can be unstable; disruptions in these third-party services can lead to execution failures. Similarly, Galaxy requires specific software and reference datasets to be available ewithin the Galaxy platform, increasing technical complexity. The platform also faces challenges in recording and exporting provenance information due to the absence of a standardised schema or vocabulary. Nextflow, while flexible, demands some technical skill from its users. Users must have some understanding of programming which requires them to specify software and hardware dependencies manually. This lack of standardisation in execution and specification complicates exploring and utilising provenance information [8]. Thus, these solutions often fall short due to data security, patient privacy, usability, and regulatory compliance concerns that are difficult to address [9]. Interoperability issues across different systems and data formats also hamper effective dataset utilisation. Moreover, existing platforms require significant computational resources, user training, and infrastructure, which may not be universally accessible to researchers, limiting widespread adoption and efficacy. Consequently, these shortcomings limit the effectiveness and widespread adoption of data-sharing-based solutions.

To address these challenges, we introduce JINet, a novel biomedical platform [10] designed for data analysis and result sharing, especially within genomic and healthcare domains. JINet provides a portfolio of self-contained applications that allow data to be analysed locally within an execution sandbox (provided by the web browser Same Origin Policy [11] and Content Security Policy [12]) to limit access to system resources, ensuring the security and privacy of sensitive information. Unlike existing biomedical platforms, which typically require data to be uploaded to remote web services, JINet runs analyses locally on local data, removing the dependency on external web services and ensuring more consistent performance. Transparently to the user, JINet locally downloads the selected analysis script and runs it within the user's web browser. In doing so, JINet removes the privacy concerns due to the risks of transferring data via potentially untrustworthy channels, by removing the need to transfer data *tout court*, sharing analysis scripts instead.

JINet aims at developing a self-sustaining community around it, composed of Application Developers, Users, and Data Providers. Developers are incentivised to integrate their latest machine Learning and Data Analysis methods into JINet, making them readily available to as many users as possible. Conversely, users are attracted by the platform's easy access to the latest tools. To make this ecosystem self-sustainable, we ensured that integrating new analysis scripts into JINet is straightforward: with a simple permission request and script submission, they are standardised by JINet for automatic access without installation requirements.

JINet provides an accessible environment for analysing healthcare datasets, empowering researchers to work with data from diverse sources without compromising confidentiality. As a self-contained platform, JINet democratises access to analysis pipelines by removing setup complexity, allowing Users with little technical proficiency to run complex analysis scripts. Moreover, JINet facilitates collaboration by enabling Users and Data Providers to share results and insights without compromising data privacy.

JINet distinguishes between Users, Application Developers, and Data Providers. Users, typically non-experts in programming, can run analyses without having to deal

with the intrinsic complexity of the application they selected and without even logging in to JINet. Workflows for these groups are illustrated in Fig. 1. Users browse the list of available applications, select the desired application, and then configure the application parameters. Application Developers can develop and make applications available to regular JINet users. Data Providers are entities that voluntarily share publicly available datasets for public use, such as benchmark datasets or examples of structured data. This kind of data might include datasets used in challenges, like the Dreamdata [13], CAFA [14], or Kaggle [15], or sample data from specific applications.

Implementation

JINet is structured as two major components: a server and browser-based application runner. The server is an index of available applications and sample data. The client, running in a web browser, executes applications on user data without transmitting these data to the server or any other party.

JINet components

Application Runner. All JINet applications run inside a web browser sandbox on the user's machine. Three runtime environments are available for Python, R, and Javascript applications. Python and R interpreters are compiled to WebAssembly. The Python interpreter is provided by Pyodide [16], and the R interpreter is provided by WebR [17]. This allows JINet to safely quarantine all data provided by users within the sandbox provided by the web browser. All network traffic to remote servers, including the JINet application distribution server, is encrypted with SSL. Outgoing requests are vetted by the browser Content Security Policy effectively preventing applications from leaking private user-provided data.

Application Distribution Server. This component is responsible for authenticating users, storage and distribution of applications, and storing encrypted results (derived data) and sample data. Users are required to authenticate to publish an application or share results. Running an application does not require a login. The primary purpose of the application distribution server is to serve as a centralised hub and index of available

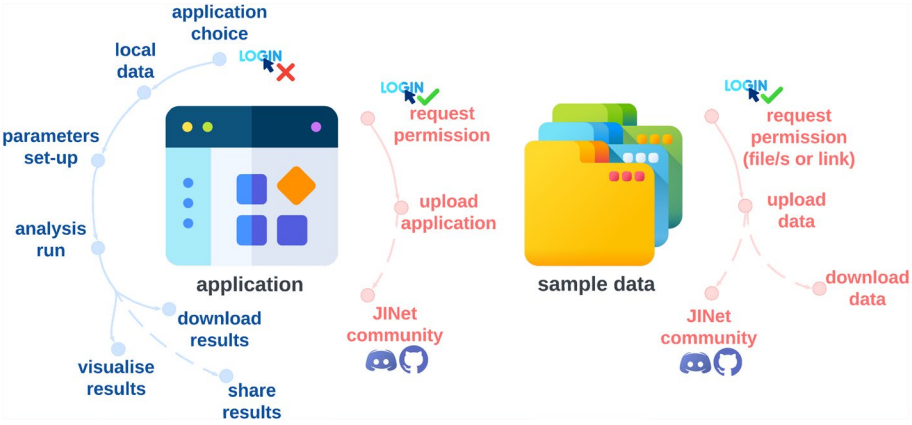


Fig. 1 Groups of users, application developers, and data providers interact with JINet in different ways. Users select an application to run, provide parameters and data, run the application, and receive results. Application developers must login, request permission to upload applications, then upload their application, and optionally participate in the community on Discord. Data providers must also login, request permission to upload, then upload sample datasets

applications and sample data. When a user selects an application to run, the application's executable representation is transferred to the user's web browser, where it is run by the Application Runner component. The application distribution server has no knowledge of when, how, or with which parameters a user has executed the application.

The user may decide to store the results of an application (e.g., a plot or an output file) on their computer. They may also choose to share the results using JINet. JINet provides a mechanism for sharing results by encrypting them in the web browser with a key derived from a passphrase and storing them temporarily on the server.

Security and privacy

While making scripts easy to use for non-experts, there are 2 negative security outcomes that JINet effectively prevents: 1) leaking user data to the internet, and 2) accessing local user data without explicit permission.

To avoid these negative outcomes, JINet runs all applications within the web browser sandbox on the user's local machine. This means that user data is never transferred over the network to run an application. This sandbox is enforced by a locked-down content security policy that whitelists only a few required web addresses that the Application Runner (running in the Users web browser) is allowed to connect to:

- https://jinet.thecolliers.xyz the JINet application itself,
- https://*.r-wasm.org for the R interpreter and pre-compiled packages,
- https://cdn.jsdelivr.net for the Python interpreter and application styling,
- https://pypi.org for the Python package installer,
- https://files.pythonhosted.org for Python packages, and
- https://raw.githubusercontent.com for hosted application sources.

Importantly, this minimises the risks of adversarial or poorly written applications leaking private user data by transferring those data over the internet to a server that they control. The web browser sandbox also effectively prevents applications from arbitrarily accessing files on the user's local file system. Any local file access must be explicitly requested and allowed by the user before applications are given access to local files.

Sharing results

Users may share results by transferring encrypted data to the JINet application distribution server. Data is encrypted in the browser client before transmission using a passphrase-derived key and the browser-provided implementation of AES-256 in CBC mode. Before encryption, JINet computes a SHA-256 checksum of the data to be shared. The encrypted data and the checksum are then transmitted to the *Server* where they are stored in a database. A unique URL is returned to the sharer which they can provide to a receiver with the decryption passphrase via a back-channel. The encrypted data and checksum are transmitted to the receiver by the *Server*. The receiver decrypts the data using the shared passphrase and *Application Runner* checks the integrity of the decrypted data based on the checksum.

Submitting an application

To submit an application to JINet, developers must write a script that defines a single entry-point function. This function serves as the primary interface for receiving and

processing all external inputs, acting as the main gateway for executing the core logic. The types of arguments that the entry-point function can accept are limited by what JINet can display user-friendly interface controls for: filesystem paths, strings, integers, floats, and booleans. Developers may download dependencies and call into other functions from this entry point. The authors expect developers to set up the necessary computational environment (e.g., read input files from the filesystem) and then execute the main script functionality from the entry-point function. Due to parameter type restrictions, complex data structures such as data frames cannot be directly passed; instead, developers should accept filenames and read the data from these files.

The entry-point function must return either an HTML string or a filename. In the case that HTML is returned, it is embedded in the web browser interface directly for display to the user. Instead, if a filename is returned, a save button is displayed, allowing users to save the resulting file.

By following these guidelines, developers can trivially integrate analysis scripts into JINet, facilitating seamless data analysis without the risks of direct data sharing.

User interface

JINet is designed with user accessibility and functionality in mind. The interface consists of several key pages that guide the user through the various features and applications available on the platform.

General Information Page: the initial landing page provides general information about JINet, offering an overview of its capabilities and purpose. This page serves as a starting point for users to familiarise themselves with the platform.

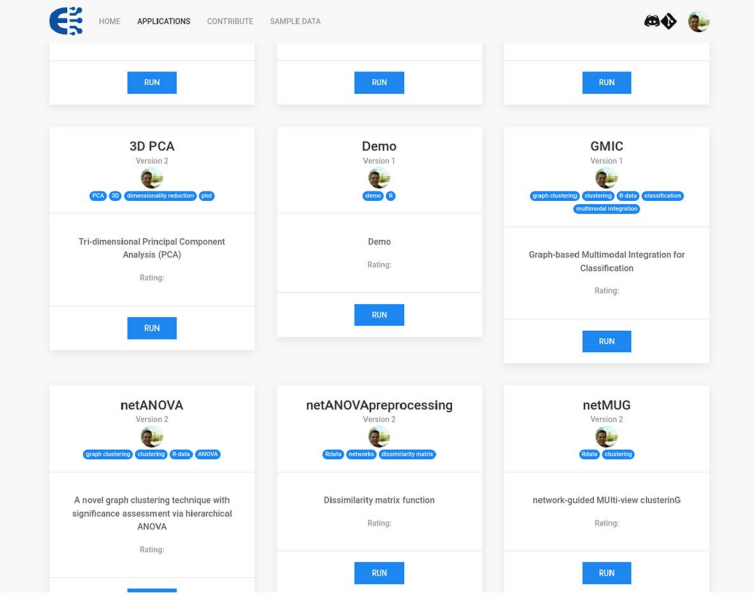
Application Selection Page: this page lists all available applications that users can choose from. The user may filter applications by tag or by text search. Each application is displayed with a brief description to help users decide which analysis to run. Notably, running an application does not require the user to log in, ensuring ease of access and use.

Upon selecting an application, the user is taken to the *Run* page for that application, where the following is displayed:

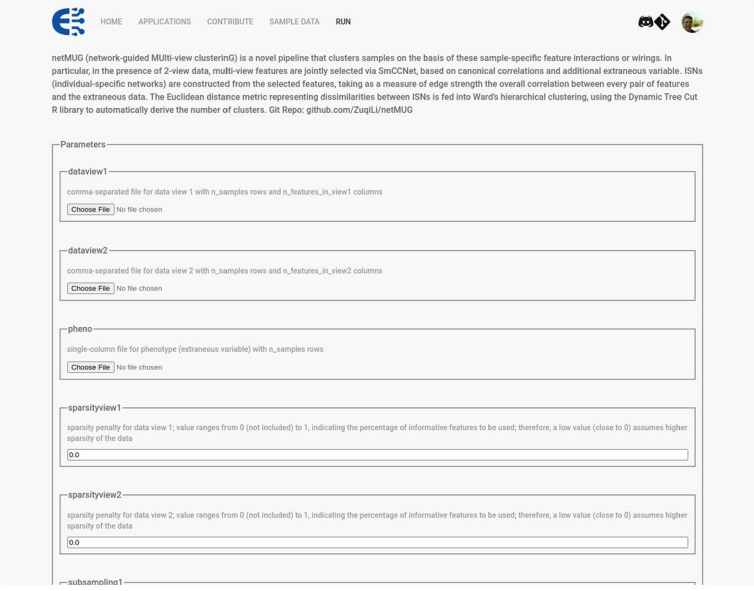
- **application name:** and the version that is running,
- **a description:** of the application itself in a longer, more detailed form than in the application list page (this can include a link to the application source code),
- **parameter inputs:** all required parameters for the application, accompanied by concise descriptions;
- **data selection:** users can select local data on which the analysis will be performed.

Once all parameters are configured, and data is selected, the user can run the application by clicking on the "Run" button. This workflow is shown in Fig. 2 for selected applications that are available at the time of writing.

Results Retrieval and Sharing: after the application completes execution, users can view the results on the web page. Where an application makes its output available as a file, a download button will be displayed, allowing the user to save the output to a file outside the browser sandbox. Additionally, JINet provides an option to share results securely. Shared results are encrypted by the web browser before being transmitted to the JINet Server. Once stored on the server the user can share a link with others. The



(a) A screenshot from JINet displaying an index of applications.



(b) Running an application is as simple as graphically selecting the parameters to run with them pressing the "Run" button.

Fig. 2 Snapshots of the current JINet interface at the time of writing

recipient can decrypt the result using a passphrase set by the user who generated the result. JINet requires users to log in to share results to minimise the risk of server storage being misused for malicious purposes.

Data Sharing and Contribution: JINet supports sample data sharing and developer contributions through dedicated forms:

- **sample data upload:** data owners can share non-sensitive versions of their datasets by logging in and requesting permission. Once granted, they can upload sample data, which becomes available for download by other users;

- **application contributions:** developers can contribute their applications to the platform. This process is similar to the data upload steps, requiring login and a permission request.

Results

Comparison with existing solutions

Table 1 outlines a comparison with other systems for running biomedical analyses. The table outlines where analysis applications are run (i.e. where user data is stored for analysis), whether result data can be shared, the relative available performance of running analysis scripts, the difficulty of setup for analysis applications, the difficulty of running analysis applications (i.e. providing correct inputs), cost, and whether there are explicit data privacy protections. For this comparison, “Users own computer” means any computer they control. This could be a laptop they own or a server provided by an institution. The descriptions assume the common usage case without considering special security constraints. For example, a user’s institution security policy may block arbitrary network requests. In these cases, we describe this as "Arbitrary network requests are allowed" since it is not the system itself enforcing a security policy.

JINet combines the functionality of a typical web service for storing applications and example datasets while offering computing capabilities directly in the user’s web browser. This hybrid approach differentiates it from other platforms that primarily operate on centralised or cloud-based infrastructure. Unlike other existing platforms (e.g., Galaxy [3], Nextflow [5], qPortal [21], or AnVIL [22]), JINet emphasises:

- **local execution for privacy**: analyses are performed entirely on the user’s local machine, ensuring that sensitive biomedical data never leaves their control. This stands in contrast to cloud-based platforms, such as AnVIL [22] and Galaxy [3], which require data to be uploaded to external servers and may raise privacy concerns;
- **accessibility for non-technical users**: JINet removes the need for software installation or complex configurations, offering an intuitive interface that lowers the barrier to entry for researchers with limited programming expertise;
- **collaborative development**: JINet fosters a community-driven ecosystem where non-experts can access cutting-edge tools, interact with developers, and contribute to the creation or refinement of applications. By enabling users to share analysis scripts and insights rather than raw data, JINet promotes secure and open collaboration.

By addressing these challenges, JINet serves as an effective alternative to existing solutions, offering unique advantages for users who prioritize data privacy, ease of use, and fostering collaboration. While systems like Galaxy [3] or Nextflow [5] excel in scalability or advanced workflows, AnVIL [22] specialises in large-scale collaborative analysis on federated datasets, JINet’s design is intentionally focused on empowering a broader audience, connecting non-experts with experts, and encouraging the open exchange of knowledge and tools.

Table 1 Comparison of JINet with existing systems for running biomedical analysis applications

System	Data location	Share results?	Compute performance	Installation/setup required to install an application
JINet	Users own computer.	Yes. Built-in.	Limited by users machine and WebAssembly [18] overhead. Single process. Single thread[1].	None
Blockchain [1]	Cloud (IPFS).	Yes. Using smart contracts.	Limited by cloud hardware.	Requires setup of Ethereum node and IPFS installation.
Galaxy [3]	Cloud	Yes. Built-in.	Limited by cloud hardware.	None
Apache Airflow [4]	Cloud	No. Not built-in. Depends on service provider.	Limited by cloud hardware.	None
Nextflow[5]	Users own computer or cloud.	No. Not built-in.	Limited by hardware it is running on. Highly parallel.	Extremely difficult for non-technical users.
Huggingface [6, 7]	Cloud	Yes. Users can share models and results with the community.	Limited by cloud hardware. Access to higher performance hardware is available to paying customers.	None
Jupyter [19]	Users own computer.	Not built-in.	Limited by users own hardware and technical skill.	User must install *at least* a compatible Python interpreter, the Jupyter package, and any kernel required by the notebook.
Google Colab [20]	Cloud	Yes. Built-in via Google Drive	Limited by cloud hardware. Subject to usage limits.	None
qPortal [21]	Cloud (openBIS).	Yes. Built-in.	Limited by cloud hardware.	Requires setup of openBIS and Liferay.
AnVIL [22]	Cloud	Yes. Built-in.	Limited by cloud hardware.	None
Manual[2]	Users own computer.	Not built-in.	Limited by users machine.	Usually difficult for non-technical users.

System	Easy to run?	Cost	Privacy	Category
JINet	Yes. Fill out a form and click run. Examples and documentation for parameters is available.	No cost	Relies on browser sandbox to prevent access to system resources. Data does not leave users computer. Arbitrary network requests are blocked by the browser sandbox.	Biomedical platform.
Blockchain [1]	Moderate.	Variable (depends on usage).	High (due to encryption and decentralisation).	Data sharing platform.
Galaxy [3]	Yes. Graphical workflows. Form inputs for applications. Network connection required.	No cost.	Data stored on the cloud is public by default. Sharing "history" is not encrypted. Limited network access.	Biomedical platform.
Apache Airflow [4]	No. Difficult for non-technical users.	Depends on cloud provider	Dependent on cloud operator.	General purpose workflow orchestration.
Nextflow[5]	No. Difficult for non-technical users	No cost when running on users own computer.	No controls. Arbitrary network requests are allowed.	General purpose workflow orchestration.
Huggingface [6, 7]	Yes.	Free (open-source).	Varies by model usage	Pre-Trained Models (PTMs) sharing platform.

Table 1 (continued)

System	Easy to run?	Cost	Privacy	Category
Jupyter [19]	No. Difficult for non-technical users.	No cost.	No controls. Arbitrary network requests are allowed.	General purpose interactive computing platform.
Google Colab [20]	No. Difficult for non-technical users	No cost for basic usage within limits. Price scales with performance requirements.	Defined by Google. Arbitrary network requests are allowed.	General purpose interactive computing platform.
qPortal [21]	Moderate (user-friendly interface).	Variable (depends on usage and resources).	High (data access regulated, user roles).	Biomedical data management platform.
AnVIL [22]	Moderate; requires understanding of cloud platforms.	Pay-per-use (cloud-based cost model).	Moderate to high (FedRAMP-certified; active threat monitoring).	Biomedical platform.
Manual	No. Can be difficult, especially for non-technical users. The user must know how to run.	No cost.	No controls. Arbitrary network requests are allowed.	General purpose computing platform.

[1] See Sect. *Temporary limitations*. [2] *Manual* here means that a user must find the analysis tool themselves, set it up on their system, and understand how to run it themselves. The comparison generalises a *class* of tools and may not be correct for a specific tool in this class

Example of usage

In this section we compare the usage of JINet compared with manually setting up and running tasks on a laptop owned by a user and doing the same process in Galaxy [3]. The task is described as follows:

> *Download the Fisher's Iris flower dataset [23] from Kaggle. Create a scatter plot with "sepal length" on the x-axis and "petal length" on the y-axis, labelled by the "target" column (the species of Iris). Ideally, the plot should have a title and the axes should be labelled.*

Set up

To use all three systems, a user must download the dataset to their laptop. Galaxy requires that the user then upload the data to the Galaxy web service. This can be achieved easily by non-technical users by pressing the download button within the Kaggle website then "unzipping" the downloaded file.

Using JINet

After browsing to the JINet web page, the user must select the application they need to complete the task. The user may select the "scatter plot" tag, or type "scatter" into the search bar to narrow the selection of applications to those for scatter-plotting data (See Fig. 3). "Scatter" stands out as potentially the correct one and the user clicks the run button where they are presented with a loading screen. The application must download 52

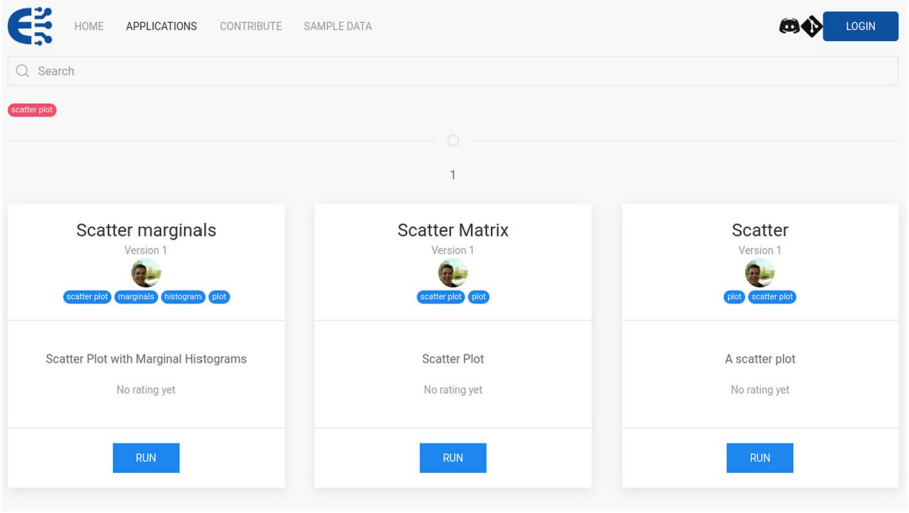


Fig. 3 A screenshot of the JINet application list page showing a filtered index of currently available applications tagged with "scatter plot"

MiB of compressed resources including a Python interpreter and any required Python modules. This takes takes approximately 30 s on a fast network or approximately 18 min on a simulated 3 G wireless network.

The user is then presented with a button asking them to select the folder where their data is (See Fig. 4a), the user selects the unzipped folder containing the Iris data and confirms that they want to allow JINet access to the files in this folder (See Fig. 4b). If this was a malicious application, it would now be able to read all of the data in the selected folder and sub-folders. But the data cannot be transferred to a web service controlled by the malicious application developer due to the web browser enforced Content Security Policy. Once JINet has access to the data folder, the application parameter input interface is displayed as in Fig. 4c. This includes a listing of the data files available in the selected data directory.

The user can now fill in the form parameters as they see fit and click on the run button. This will execute a Python plotting script in WebAssembly on a worker thread. Once the script has finished executing, it returns a string of HTML which is displayed as an interactive plot of the input data as seen in Fig. 5a.

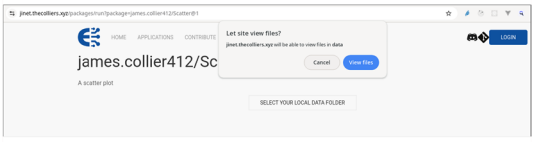
Manually completing the task

The user must write a plotting script for themselves or find a script already available elsewhere. In the case that they find the Scatter application script on the JINet git repository (available from https://github.com/GiadaLalli/JINet/blob/main/doc/examples/scatter.py), they must download this script, install a Python interpreter and the required packages that the script needs. They must then understand the script enough to determine the command line arguments that are required.

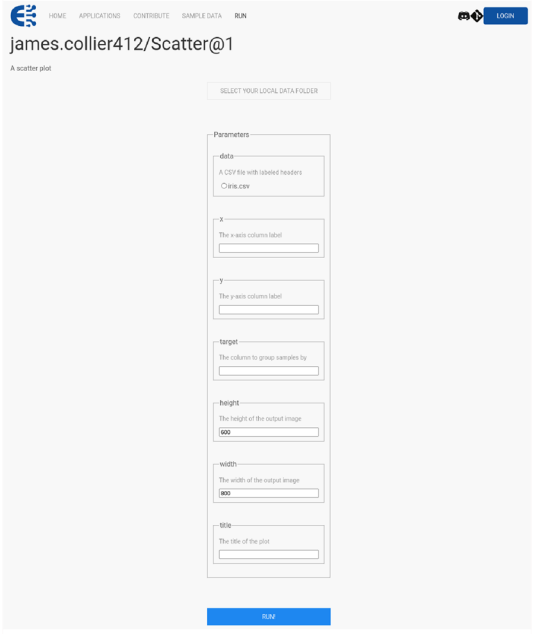
When they run the script there is likely[1] nothing preventing a malicious script from sending the input data or any other data available on the laptop to a server controlled

[1] Unless the user has taken explicit precautions to prevent malicious activity such as running the script in a sandbox and disabling any network connections.

(a) The user must first select the local folder where JINet can access user data.



(b) The web browser asks the user to confirm that JINet should be given permission to view files in the selected folder.



(c) Once JINet has access to the local data, the remaining application parameters are shown with standard input controls.

Fig. 4 A series of screenshots illustrating the usage of the parameter interface provided by JINet for the "Scatter" demo application

by the malicious script developer. This is not possible when running the same malicious application within JINet.

In this case, running the script produces a static, non-interactive image (See Fig. 5b). The image output is produced much faster than with JINet or Galaxy. However, in order to adjust the image produced, the user needs to be able to understand the programming language and the libraries used enough to make modifications to the script. This is in stark contrast to JINet and Galaxy interfaces, where adjusting parameters only requires changing values with intuitive and familiar user interface controls.

Using Galaxy

Galaxy has a number of very useful properties overlapping with JINet. There are many Galaxy instances with free access in contrast to a single instance provided by JINet. Users do not need to login to run individual tools. The user must trust the administrators running the instance to preserve the privacy of their data, although the tools provided by Galaxy are likely to be well vetted by the community.

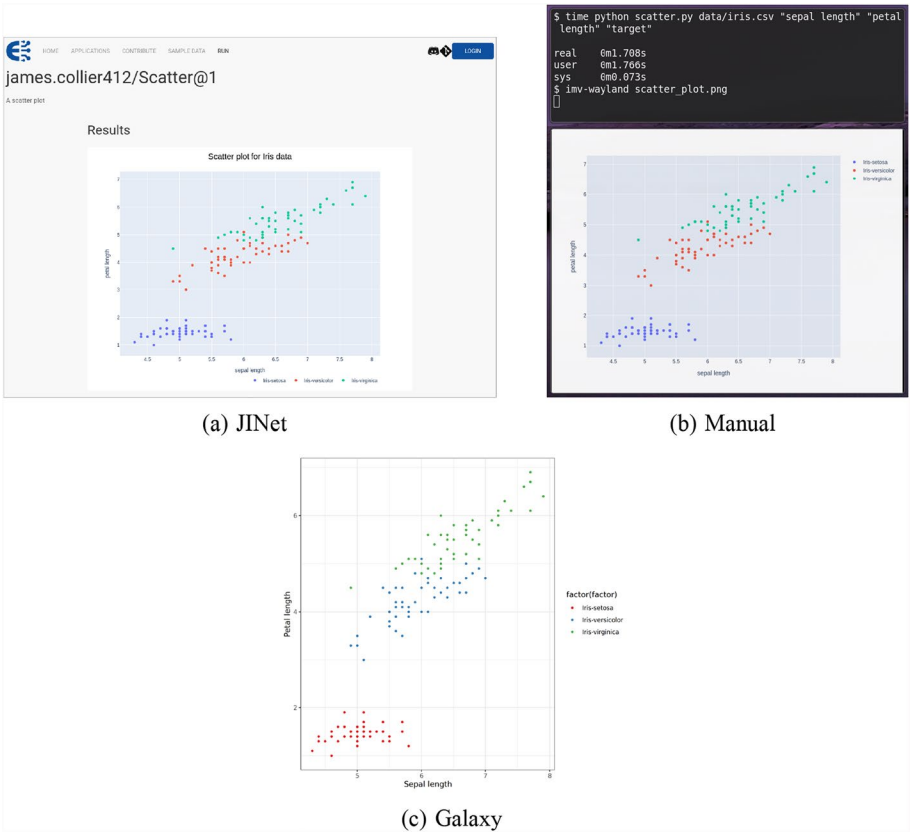


Fig. 5 Results produced by each of **a** JINet **b** Manual, and **c** Galaxy when plotting Iris data with "sepal length" on the y-axis and "petal width" on the x-axis

After navigating to a Galaxy instance, the user must select the appropriate tool. Searching for "scatter" narrows the selection of tools and "Scatterplot with ggpot2" is one of applications that a user might choose. After clicking on the chosen application the user is presented with intuitive and familiar user interface controls to run the plotting tool (See Fig. 6).

When the user clicks "Run tool", the task is scheduled. Sometime later the resulting plot becomes available as a static non-interactive image. This is in contrast to JINet where the plot can be interactive.

Discussion

This section discusses the practical implications, limitations, and future directions of JINet. We categorise the limitations into fundamental and temporary constraints, highlight use cases outside the scope of JINet's current capabilities, and evaluate its performance on a representative real-world biomedical application.

Fundamental limitations

JINet applications run within a web browser process. This makes JINet unsuitable for distributing computations on a compute cluster. The filesystem is represented in memory on non-Chromium-based browsers, so large files may cause the application to be stopped prematurely. Whether this is a fundamental limitation depends on the standardisation of native filesystem access within web browsers.

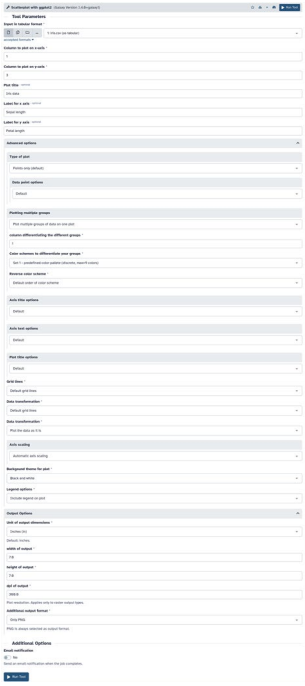


Fig. 6 A screenshot of the "scatterplot with ggplot" visualisation parameters provided by Galaxy

Temporary limitations

JINet can still improve security, performance, and usability. There is no in-principle reason the application web address allow-list should not be empty (not allowed to access any URL; except the JINet application distribution server) so with future engineering effort, JINet can become even more resistant to leaking user data. Where applications require extra files in order to function properly, these can be provided by JINet.

The relatively poor performance of numerical compute workloads (see Supplementary material) should disappear with time when high-performance BLAS libraries are packaged. Language interpreters are also continuously improving performance with an expectation that the performance gap between native and WebAssembly interpreters should be around $2\times$–$2.5\times$ [24].

There are remaining engineering tasks to make the platform more streamlined and easy-to-use. These include friendly error reporting, user interface improvements for users and application developers. Future updates could extend the supported language runtimes to include Julia and other languages, further broadening its applicability and versatility.

Another important limitation lies in assisting users with data that does not meet the requirements of an application they wish to use. JINet currently relies on users to ensure their datasets meet these requirements, or to identify the need for preprocessing before running analyses. This limitation is mitigated in two ways. Each JINet application may be accompanied by a sample dataset illustrating the required input format to allow users to compare their data against a template and determine whether preprocessing is necessary. Additionally, users can request community assistance or tailored preprocessing applications by engaging with the community via the JINet Discord channel or the GitHub repository. While JINet does not yet offer a comprehensive suite of

preprocessing applications, data validations, or pipelining tools, these would be a very useful objective for future development.

Federated learning and distributed computation

JINet is not a platform for Federated Learning [25] or distributed computation [26]. These use cases address different challenges compared to JINet's current objectives. While supporting such use cases would be an interesting enhancement, JINet is currently focused on enabling users to securely and easily run analysis scripts locally.

Real-world application

The "Gradient Boosting regression" demo application was created to demonstrate JINet on a realistic biomedical dataset and problem. This application is based on the Gradient Boosting regression example from the Scikit-learn [27] documentation which uses a real-world diabetes dataset [28]. This application produces plots of *Deviance*, *Feature Importance*, and *Permutation Importance* when predicting diabetes disease progression (the target numerical column). Maintaining the default parameters, this application runs in approximately 5 s within JINet and natively on the benchmark machine described in the Performance section of the Supplementary Material.

Conclusions

JINet is a platform designed to democratise access to effective data analysis tools while addressing critical challenges in data privacy, usability, and interoperability. JINet allows users to perform analyses without requiring software installation, configuration, or dependency management. By running analyses locally in a web browser, JINet ensures that sensitive data never needs to be transmitted to potentially untrustworthy servers, thereby prioritizing data security and user privacy.

For application developers, JINet provides a streamlined mechanism to publish robust analysis tools without the overhead of hosting and managing their own servers or developing complex user interfaces. The platform fosters a collaborative ecosystem where developers can easily integrate new methods, making them accessible to a broad audience, including researchers with limited technical expertise. JINet demonstrates that interoperability can be practically achieved while maintaining accessibility and data integrity/security.

Application developers have often turned to the Shiny [29] library for *R* to create user-friendly, web-based interfaces for their applications. Shiny operates by running on a web server where data is processed and generating HTML and JavaScript that is rendered in a user's web browser. However, this approach has notable limitations. When the Shiny server is hosted on the internet, users are required to upload their data to the server, which raises privacy and security concerns. Alternatively, users can choose to download the application to run it locally, but this requires software installation and configuration. Even if the application is distributed as a container image, users must still set up a container runtime environment, adding to the complexity.

In contrast, JINet eliminates these barriers by requiring only a web browser—an application ubiquitous across devices ranging from smartphones and workstations to televisions. With JINet, users avoid the need for installations, server dependencies, or runtime

configurations, simplifying the process and ensuring their data remains securely on their local devices.

JINet addresses significant limitations in existing platforms such as Galaxy [3], Apache Airflow [4], and Nextflow [5]. These platforms often require substantial computational resources, user training, and infrastructure, which can create barriers to adoption. JINet, in contrast, is lightweight and intuitive, minimising these obstacles. By eliminating reliance on third-party web services and cloud-based storage, JINet avoids common pitfalls such as execution instability and privacy concerns while enabling seamless integration of diverse data formats and tools.

A key advantage of JINet is its ability to balance computational scalability with privacy, usability, and accessibility. Unlike platforms like Galaxy [3], which present different trade-offs between complexity, computational demands, and security, JINet prioritises data privacy and ease of use. This trade-off positions JINet as a unique tool for addressing critical challenges in modern healthcare research, including limited dataset accessibility and interoperability issues. By achieving these objectives, JINet represents a strategic step toward advancing medical research through secure, transparent, and user-friendly data analysis solutions.

Abbreviations

SSL	Secure sockets layer
AES	Advanced encryption standard
CBC	Cipher block chaining
SHA	Secure hash algorithm
URL	Uniform resource locator
HTML	Hypertext markup language
BLAS	Basic linear algebra subprograms
WASM	WebAssembly

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-025-06244-8.

Supplementary Material 1.

Acknowledgements
The authors wish to thank designer Lorenzo Casari for the design of the JINet logo.

Author contributions
GL: Original idea, Design, Software development, Application collection, Visualisation, Writing: Original Draft, Review & Editing; JC: Design, Software development, Validation, Visualisation, Writing: Original Draft, Review & Editing; YM: Supervision, Writing: Review & Editing; DR: Writing: Original Draft, Review & Editing, Supervision, Project Administration, Resources, Funding Acquisition.

Funding
DR is funded by a FWO senior post-doctoral fellowship, Grant Number 12Y5623N.

Data availability
The "Iris" dataset used in the workflow comparison is available from this URL: https://www.kaggle.com/datasets/uciml/iris. The diabetes dataset used in the "Real-world application" section is available from the following URL: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html or importable from the scikit-learn Python module. The source code for JINet is available from our Git Repository: https://github.com/GiadaLalli/JINet. ***Project name***: JINet. ***Project home page***: https://jinet.thecolliers.xyz. ***Operating system(s)***: Platform independent. ***Programming language***: Python and JavaScript. ***License***: Apache—2.0. ***Any restrictions to use by non-academics***: none.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 14 September 2024 / Accepted: 30 July 2025
Published online: 16 October 2025

References

1. Jianjun S, Ming L, Jingang L. Research and application of data sharing platform integrating ethereum and ipfs technology. In: 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), 2020;pp. 279–282. IEEE.
2. Wang R, Tsai W-T, He J, Liu C, Li Q, Deng E. A medical data sharing platform based on permissioned blockchains. In: Proceedings of the 2018 International Conference on Blockchain Technology and Application, 2018;pp. 12–16.
3. The Galaxy Community. The galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. Nucleic Acids Res 2024.
4. Apache Airflow. https://airflow.apache.org/. Accessed: 2024-12-20.
5. Nextflow. https://www.nextflow.io/. Accessed: 2024-12-20.
6. Huggingface. https://huggingface.co. Accessed: 2024-12-20.
7. Jones J, Jiang W, Synovic N, Thiruvathukal G, Davis J. What do we know about hugging face? A systematic literature review and quantitative validation of qualitative claims. In: Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2024;pp. 13–24.
8. Cohen-Boulakia S, Lemoine F. Workflows for bioinformatics data integration. Biological Data Integration: Computer and Statistical Approaches, 2024;pp. 53–85.
9. ...Dove ES, Joly Y, Tassé A-M, Burton P, Chisholm R, Fortier I, Goodwin P, Harris J, Hveem K, Kaye J, Kent A, Knoppers BM, Lindpaintner K, Little J, Riegman P, Ripatti S, Stolk R, Bobrow M, Cambon-Thomsen A, Dressler L, Kato K, Rodriguez LL, McPherson T, Nicolás P, Ouellette F, Romeo-Casabona C, Sarin R, Wallace S, Wiesner G, Wilson J, Zeps N, Simkevitz H, De Rienzo A, Knoppers BM. Public Population Project in Genomics, Society (P3G) International Steering Committee, International Cancer Genome Consortium (ICGC) Ethics, and Policy Committee. Genomic cloud computing: legal and ethical points to consider. Eur J Hum Genet. 2015;23(10):1271–8.
10. Navale V, von Kaeppler D, McAuliffe M. An overview of biomedical platforms for managing research data. J Data Inf Manag. 2021;3(1):21–7.
11. Same Origin Policy. https://www.w3.org/Security/wiki/Same_Origin_Policy. Accessed: 2024-12-20.
12. Stamm S, Sterne B, Markham G. Reining in the web with content security policy. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, page 921–930, New York, NY, USA, 2010. Association for Computing Machinery.
13. Dreamdata. https://dreamdata.io/. Accessed: 2024-12-20.
14. Cafa. https://biofunctionprediction.org/cafa/. Accessed: 2024-12-20.
15. Kaggle. https://www.kaggle.com. Accessed: 2024-12-20.
16. The Pyodide development team. pyodide/pyodide, August 2021.
17. Stagg GW, Lionel H, and Others. webR: The statistical language R compiled to WebAssembly via Emscripten, November 2023.
18. Andreas Rossberg. WebAssembly Core Specification. Technical report, W3C, 2019.
19. Granger BE, Pérez F. Jupyter: thinking and storytelling with code and data. Comput Sci Eng. 2021;23(2):7–14.
20. Google. Google colaboratory. https://colab.research.google.com/, 2024. Accessed: 2024-12-20.
21. Mohr C, Friedrich A, Wojnar D, Kenar E, Polatkan AC, Codrea MC, Czemmel S, Kohlbacher O, Nahnsen S. qportal: a platform for data-driven biomedical research. PLoS ONE. 2018;13(1):e0191603.
22. Schatz MC, Philippakis AA, Afgan E, Banks E, Carey VJ, Carroll RJ, Culotti A, Ellrott K, Goecks J, Grossman RL, et al. Inverting the model of genomics data sharing with the nhgri genomic data science analysis, visualization, and informatics lab-space. Cell Genom 2(1) 2022.
23. Fisher RA. The use of multiple measurements in taxonomic problems. Ann Eugen. 1936;7(2):179–88.
24. Jangda A, Powers B, Berger ED, Guha A. Not so fast: Analyzing the performance of WebAssembly vs. native code. In: 2019 USENIX Annual Technical Conference, pages 107–120, 2019.
25. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RGL, Eichner H, El Rouayheb S, Evans D, Gardner J, Garrett Z, Gascón A, Ghazi B, Gibbons PB, Gruteser M, Harchaoui Z, He C, He L, Huo Z, Hutchinson B, Hsu J, Jaggi M, Javidi T, Joshi G, Khodak M, Konecný J, Korolova A, Koushanfar F, Koyejo S, Lepoint T, Liu Y, Mittal P, Mohri M, Nock R, Özgür A, Pagh R, Qi H, Ramage D, Raskar R, Raykova M, Song D, Song W, Stich SU, Sun Z, Suresh AT, Tramèr F, Vepakomma P, Wang J, Xiong L, Xu Z, Yang Q, Yu FX, Yu H, Zhao S. Advances and open problems in federated learning. Foundations and Trends® in Machine Learning. 2021;14(1—-2):1–210.
26. Andrew S. Tanenbaum and Maarten van Steen. Distributed Systems. distributed-systems.net, 3 edition, 2017.
27. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
28. Efron B, Hastie T, Johnstone I, Tibshirani R. Least angle regression. Ann Stat. 2004;32(2):407–99.
29. Chang W, Cheng J, Allaire JJ, Sievert C, Schloerke B, Xie Y, Allen J, McPherson J, Dipert A, Borges B. shiny: Web Application Framework for R, 2024.

Publisher's Note

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-025-06244-8.